An annotated overview of dynamic network flows

Balázs Kotnyek

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An annotated overview of dynamic network flows

Balázs Kotnyek

N° 4936

Septembre 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4936--FR+ENG

INRIA
SOPHIA ANTIPOLIS

An annotated overview of dynamic network flows*

Balázs Kotnyek†

Thème 1 — Réseaux et systèmes
Projet Mascotte

Rapport de recherche n° 4936 — Septembre 2003 — 28 pages

Abstract: The need for more realistic network models led to the development of the dynamic network flow theory. In dynamic flow models it takes time for the flow to pass an arc, the flow can be delayed at nodes, and the network parameters, e.g., the arc capacities, can change in time. Surprisingly perhaps, despite being closer to reality, dynamic flow models have been overshadowed by the classical, static model. This is largely due to the fact that while very efficient solution methods exist for static flow problems, dynamic flow problems have proved to be more difficult to solve. Our purpose with this overview is to compensate for this eclipse and introduce dynamic flows to the interested reader. To this end, we present the main flow problems that can appear in a dynamic network, and review the literature for existing results about them. Our approach is solution oriented, as opposed to dealing with modelling issues. We intend to provide a survey that can be a first step for readers wondering whether a given dynamic network flow problem has been solved or not. Besides restating the problems, we also describe the main proposed solution methods. An additional feature of this paper is an annotated list of the most important references about the subject.

Key-words: Flows, dynamic networks, time-dependent flows, algorithms, complexity

* This work was partially supported by the European FET project CRESSCO and the French CNRS AS Dynamo.
† I3S & INRIA Sophia Antipolis. The author received a scholarship from the European RTN project ARACNE.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Un état de l’art commenté sur les réseaux de flots dynamiques

Résumé : Un réseau de flots est dit dynamique si la traversée des arcs prend du temps, si le flot peut être retenu dans les nœuds, ou encore si les paramètres du réseau, par example, la capacité des arcs, peut évoluer au cours du temps. Dans ce rapport nous proposons un état de l’art des principaux problèmes liés aux réseaux de flots dynamiques, qui se veut une introduction au domaine. Nous y présentons les problèmes, les principales méthodes proposées pour les résoudre et nous fournissons une liste annotée des principales références bibliographiques sur le sujet.

Mots-clés : Flots, résaux dynamiques, flots dynamiques, algorithmes, complexité

1 Introduction

Imagine that we take on the difficult task of determining how many cars the streets of a town can serve. We model the street system as a network, and assign capacities to the edges representing the number of lanes on the streets. Then we can apply a network flow algorithm to determine the maximum traffic between any given pair of points. Did we answer the question? No, what we have found is the maximum number of cars that can pass the streets in a single wave, i.e., if the cars all leave at the same time and get to any point with no delay. We would rather need a more realistic model in which it takes time to travel and cars can delay their departure. It would also be desirable to include the effect of traffic jams by changing the capacity of streets that become blocked. Answering such questions is the goal of the dynamic network flow problems.

Static network flow problems have been in the focus of interest for many years and they represent a very successful area of combinatorial optimization. Deep theorems and efficient algorithms have been developed. Not surprisingly, dynamic flow problems are significantly more difficult to tackle. First of all, there is a wider range of possible problems that do not appear in static networks. For example, one can minimize the time needed for sending a given amount of flow between two points. Furthermore, dynamic problems are more difficult to solve than the static ones. Several of the arising problems are NP-hard. The differences from the classical problems made it necessary to devise new techniques, although most of the solution methods eventually reduce the dynamic problem to a static one and then employ existing algorithms.

The purpose of this survey paper is to give a description of problems and methods in dynamic networks. It is intended to be a starting point for readers who are new to the area and would like to find out what has been done, which questions are settled and which are not, or where to look for further references. To this end, we briefly review the basic concepts of static and dynamic networks (in Section 2 and 3), list the important problems that can be addressed in a dynamic model (Section 4), and then point out the main solution approaches (Section 5). We do not describe the solution methods in detail but we always give references where the interested reader can find the points not expounded here. In Section 6 we give a short description of the papers used to compile this survey, which can help the readers to find the reference relevant to their problem. We close the paper by pointing out a possibility for further research.

Dynamic networks were introduced in 1958 by Ford and Fulkerson [16, 17]. They dealt with the maximum flow problem in discrete time and developed a technique that is still widely used. Since then, several further problems have been analyzed, such as, for example, the quickest, minimum cost or earliest arrival flows, continuous-time settings, or models where the parameters change with time. Previous surveys on general dynamic flow problems include those by Aronson [4], Lovetskii and Melamed [28], and Powell, Jaillet and Odoni [38]. The earliest of these three, [28], provides a good description of several problems and techniques in both discrete and continuous time. Aronson [4] concentrates on the maximum flow and transshipment problems in discrete time. Its main advantage is the extensive coverage of applications. The focus of [38] is on modelling issues, like for example that of the time horizon. It deals only with discrete-time settings but mentions problems where the parameters are not fixed.

Some words about terminology should be added here. A few authors (e.g., Fleischer in [10]) argue that the word 'dynamic' is more consistently used for a problem with input that changes over time. In a dynamic flow problem it is the solution that changes, the network is fixed or its changes are known in advance. Therefore these authors prefer to use the terms *flow over time* or *time-dependent flow*. On the other hand, it is widely accepted (as expressed, e.g., in [38]) that a problem is called dynamic if one or more of its parameters is a function of time. This can involve problems where the input is constantly changing or where the input is time-dependent but known in advance. We also subscribe to this opinion and use the more compact 'dynamic flow' term in this paper.

2 Static flows

In this section we give an overview of the most important results about static flows, which are relevant in the further discussion about dynamic flows. We assume that the reader is familiar with the subject, therefore we let ourselves be succinct. A comprehensive description of static flow problems and solution methods can be found in [1].

A static flow is defined on a directed graph $G = (V, A)$. The capacity of an arc a is denoted by u_a, the cost (if any) per unit flow on a is c_a. The set of arcs entering and leaving node v are denoted by $\delta^+(v)$ and $\delta^-(v)$, respectively. There is a source r and a sink s in the graph. In the *maximum flow problem* the aim is to find the maximum flow between the source and the sink:

$$\begin{aligned}
&\text{maximize} \quad d \\
&\text{subject to} \sum_{a\in\delta^+(r)} x_a - \sum_{a\in\delta^-(r)} x_a = -d && (1)\\
&\sum_{a\in\delta^+(v)} x_a - \sum_{a\in\delta^-(v)} x_a = 0 \qquad \forall v \in V \setminus \{r,s\} && (2)\\
&\sum_{a\in\delta^+(s)} x_a - \sum_{a\in\delta^-(s)} x_a = d && (3)\\
&0 \le x_a \le u_a \qquad \forall a \in A && (4)
\end{aligned}$$

A set of non-negative arc values $x : A \to \mathbb{R}_+$ is a *flow* if it satisfies the flow conservation rule (2). A flow is feasible if it complies with the capacity constraints (4). A value d that is conform with constraints (1) and (3) is the *value of the flow*, usually denoted as $|x|$. A flow with value 0, i.e. a flow that satisfies the flow conservation rules at every node, is called a *circulation*. A flow with value d can be easily transformed to a circulation by adding an extra arc from s to r carrying a flow of d.

An alternative formulation is when the flow is decomposed to *chains*. Let $\mathcal{P}$ be the set of directed paths (called chains) from r to s. A flow $f(P)$ on a chain $P \in \mathcal{P}$ has a positive value $|f(P)|$ on the arcs of P (these arcs are denoted as $a \in P$), and 0 elsewhere. In chain flows the flow conservation at

the nodes is satisfied by the construction. The maximum flow problem is then formulated as follows.

$$\begin{aligned} \text{maximize} \quad & d \\ \text{subject to} \quad & \sum_{P \in \mathcal{P}} |f(P)| = d \\ & \sum_{a \in P} |f(P)| \leq u_a \qquad \forall a \in A \end{aligned}$$

Circulations can be decomposed to chain flows and flows on cycles. Obviously, there is no need for cycles in a flow from r to s.

In the *minimum cost flow problem* the flow value d is fixed, and we are looking for a flow that has this value and achieves it with minimal cost:

$$\text{minimize} \quad \sum_{a \in A} c_a x_a \qquad \text{subject to (1)} - \text{(4)}$$

2.1 Variations of the basic model

In a version of the basic network flow problem there are multiple sources and multiple sinks, and one, for example, tries to maximize the total flow from the sources to the sinks. It can be easily reduced to a single-source single-sink problem by introducing a supersource, a supersink, and artificial arcs with infinite capacity to connect them to the sources and the sinks, respectively. Now the flow conservation rule must hold at the original source and sink nodes, therefore the total amount of flow that leaves the original sources equals the flow that leaves the supersource.

A modification of this is when fixed supplies at the sources and demands at the sinks are given, and we ask for a feasible flow that empties the supplies and satisfies the demands. This problem is called the *transshipment problem*. It can also be reduced to a single-source single-sink problem by introducing a supersource and a supersink and connecting them to the original sources and sinks, but now with arcs whose capacities equal the supply or demand of the node they are connected to. There exists a feasible transshipment if and only if the value of the maximum flow in the transformed network is the sum of the supplies (which should be the same as the sum of the demands).

The flow problems so far were single-commodity flows. Multicommodity problems arise when several flows use the same network. They may have different sources and sinks, they satisfy the flow conservation rule separately, but are bound together by the common capacity constraints. Multicommodity problems are much harder to solve than single-commodity problems. For example, the integral multicommodity problem, where we look for integral flows, is NP-hard.

2.2 Complexity

As a summary of this section, we remind the reader of the complexity of the basic static flow problems. A thorough description can be found in [1].

The maximum flow problem and the problems that can be easily reduced to it, are polynomially solvable. Thus there is a polynomial algorithm to find the maximum flow with single or multiple

sources and sinks, or to find a feasible transshipment. Similarly, the minimum cost flow or circulation can be found in polynomial time for linear or convex cost functions. The same holds for the problem of finding integer solutions. The fractional multicommodity problem is polynomially solvable, while the integer version is NP-hard.

3 The basic dynamic network flow models

In a dynamic network flow model there is an additional parameter τ_a assigned to each arc a measuring the time a unit flow takes to get from the tail of arc a to its head. In other words, a flow unit that leaves the tail of a at time t arrives at the head at time $t + \tau_a$. We will call τ_a the *traversal time* or *transit time* of arc a.

3.1 Time horizon

We differentiate between models with finite and infinite *time horizons*. In the finite horizon models everything must happen before a given stop time T. The stop time is chosen so that the relevant flow units can arrive before it, in other words, no flow can enter an arc that will not arrive before T. Most of the theoretical and practical models use finite time horizon. Infinite time horizon, however, appears in a number of practical problems. For example, traffic assignment problems can be modeled with a rolling horizon, where activities are initiated endlessly but the optimization focuses on given periods, say days. Another infinite horizon model is where the activities happen periodically, like for example in inventory problems. If the cost of the flow is concerned, then one can discount future costs and calculate the present value. More about modelling questions for infinite time horizon can be found in [38]. Orlin [34, 35] introduced the concept of 'throughput' in dynamic networks with infinite time horizons (basically, the flow circulating in the network) and provided theoretical and practical results for the maximum throughput and the minimum cost fixed throughput problems. In this paper we deal with models with finite time horizon.

Time can pass in discrete increments or continuously. In discrete-time models we can assume without loss of generality, by choosing a suitable unit, that we look at the network at times $t = 0, 1, 2, \ldots, T$ and all time-related parameters are integers. We will see that this is the easier case; a continuous-time problem, where t can take any value in $[0, T]$, is a harder nut to crack. In practical models time can be discretized, thus converting continuous flow models to discrete ones. Obviously, the choice of the time unit has a great impact on the complexity of the problem, the shorter the time limit is, the more complex the problem becomes. To be able to use general notations that are valid for both discrete and continuous models, we will denote the time domain by $\mathcal{T}$. Thus $\mathcal{T} = \{0, 1, 2, \ldots, T\}$ in a discrete model and $\mathcal{T} = [0, T]$ in a continuous model.

In more general dynamic flow models the parameters of the arcs are not fixed but varying with time. Thus, we have functions $u_a : \mathcal{T} \to \mathbb{R}_+$, $c_a : \mathcal{T} \to \mathbb{R}$, $\tau_a : \mathcal{T} \to \mathbb{R}_+$ to denote the capacity, cost and traversal time of an arc a at different times.

3.2 Load, flow and throughput

In static models the flow that enters an arc is indistinguishable of the flow that is on the arc, as everything that enters will leave instantly. This is not the case in dynamic models, where the flow spends time on an arc, so a flow unit that entered earlier might easily be still on the arc. Hence the difference between the *load*, i.e., the flow on the arc; and the *flow rate*, the amount of flow that enters the arc. It is the flow rate that is in the focus of interest, and its name is usually shortened to 'flow'. Both are functions of time. Let $f_a : \mathcal{T} \to \mathbb{R}_+$ and $x_a : \mathcal{T} \to \mathbb{R}_+$ denote the load and the flow rate on arc a, so that $f_a(t)$ is the load on the arc at time t, $x_a(t)$ is the amount of flow that enters the arc at time t. These functions are strongly related, loosely speaking the load is the integral of the flow rate and the flow rate is the first derivative of the load. More formally, in continuous and discrete model, respectively, we have:

$$f_a(t) = \int_0^{\tau_a} x(t-\theta)\,\mathrm{d}\theta, \qquad \text{and} \qquad f_a(t) = \sum_{\theta=0}^{\tau_a - 1} x(t-\theta)$$

Note that it is up to convention that the flow entering arc a at $t - \tau_a$ has already arrived at the head node by t or is still on the arc at that moment. As it is clear from the discrete formula above, we assume that such a flow is already at the head node at time t.

Recall that in the street traffic system the capacity of a street was given by the number of the lanes. That is, we gave a limit on the number of the cars that can enter the street at the same time. In dynamic models it is usually the flow rate that is bounded by the capacity of an arc. A limit on the load can be calculated from this capacity. For example, an arc with capacity 2 and traversal time 3 can have a load up to 6 at any time.

The most important measure of a dynamic network flow is its *throughput*, the amount of flow that can be sent from the source to the sink in the given time interval. A simple, illustrative version is the following: How much flow can arrive in time interval $[0, 2\tau)$ from the tail to the head of arc a whose capacity is u and traversal time is τ? Obviously, any flow entering the arc after τ will not arrive before 2τ, so we ignore it. The maximum flow that can be sent is u, so let $x_a(t) = u$ for $t \in [0, \tau)$ and 0 for $t \in [\tau, 2\tau)$. The throughput in $[0, 2\tau)$ on arc a is the integral of the flow rate over time:

$$\int_0^{2\tau} x_a(t)\,\mathrm{d}t = \int_0^{\tau} u\,\mathrm{d}t + 0 = \tau u.$$

If the time is discrete, so τ is integer and we ask for the throughput in $[0, 2\tau - 1]$, any flow entering the arc later than $\tau - 1$ will not arrive in time. We can send at most u units at times $0, 1, \ldots, \tau - 1$, so the total throughput is τu again.

There is a *natural transformation* of discrete flows to continuous ones. Let $x_a(t)$ be a feasible discrete flow entering arc $a \in A$ at time t ($t = 0, 1, \ldots, T-1$) and set the continuous flow rate $y_a(\theta)$ to $x_a(t)$ for $\theta \in [t, t+1)$. If we assume that the arc capacities do not change, then we get a feasible continuous flow with this transformation, and the throughput in any integral interval $[t, t+k)$, $t = 0, 1, \ldots, T-1$, $k \in \mathbb{N}$ will be the same for both x and y. Fleischer and Tardos [15] showed that for many problems this natural transformation preserves optimality. Thus, the

continuous version of these problems is not harder than the discrete one: an optimal discrete solution yields an optimal continuous solution via the natural transformation.

3.3 An example

We give an example of dynamic flows. Consider the network in Figure 1. The arc labels are the traversal times, every arc has capacity 2. In Figure 2 we give a flow that has a throughput of 8 in $\mathcal{T} = \{0, 1, \ldots, 5\}$. The diagrams give snapshots for the unit periods. The values given are the amounts of flow sent, the bold parts of the arcs represent the portion these flow units take during the time steps. For example, 2 units of flow enters the two arcs starting from r at $t = 0$. One of them accomplishes the third of the arc in one unit time, the other arrives at v at $t = 1$, and then it is divided up to two unit flows going towards u and s.

3.4 Storage at nodes

It may be necessary that the flow waits at a node until it can continue on an arc. In other words, nodes have a *storage* (or *holdover*) capability. Let b_v denote the fixed capacity of storage at node v, or $b_v : \mathcal{T} \to \mathbb{R}_+$ if the capacity is a function of time. In some important cases the storage at intermedial nodes is assumed but not used, so in this case b_v can be set to 0 for all $v \in V \setminus \{r, s\}$. There are problems (e.g., in [14]), where storage is explicitly not allowed in nodes other than the source and the sink. The default assumption is that storage is not limited, we point out only the models where this is not true.

If storage at nodes is constrained, then unexpected things may happen. For example, a maximum flow may use cycles. To see why, imagine a network where a bottleneck arc is temporally unavailable, so it makes sense for the flow to spend time by cycling until the arc becomes passable again. If the storage is not restricted then this does not happen, as flow can wait at nodes instead of cycling around.

Storage means that the flow conservation is not satisfied at each time instance because the amount of flow arriving at a node at a given time can be different from the amount of flow that leaves the node at that time. To overcome this, two techniques are known. The first, which works only in discrete-time models, introduces artificial loops at the nodes with unit traversal time, and the storage is imagined as flow circulating on the loops. The other technique notices that for each possible time instance θ, the amount of flow that has arrived at a node by θ cannot be more than the amount of flow that has left the node before θ. Thus the flow conservation constraints are summed over time and satisfied with inequality instead of equality. It is usually also assumed that there is no flow left in the network at T, so the flow conservation constraints are satisfied with equality for $\theta = T$.

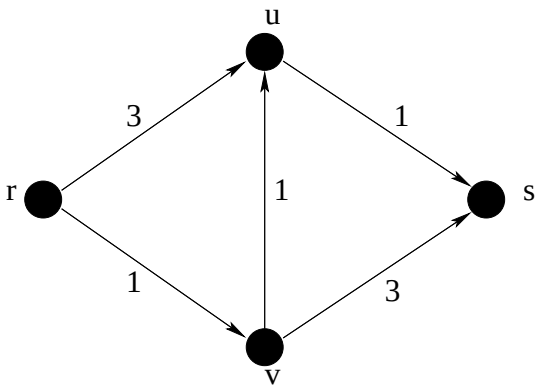

Figure 1: An example dynamic network. The arc labels are the traversal times. All capacities are 2.

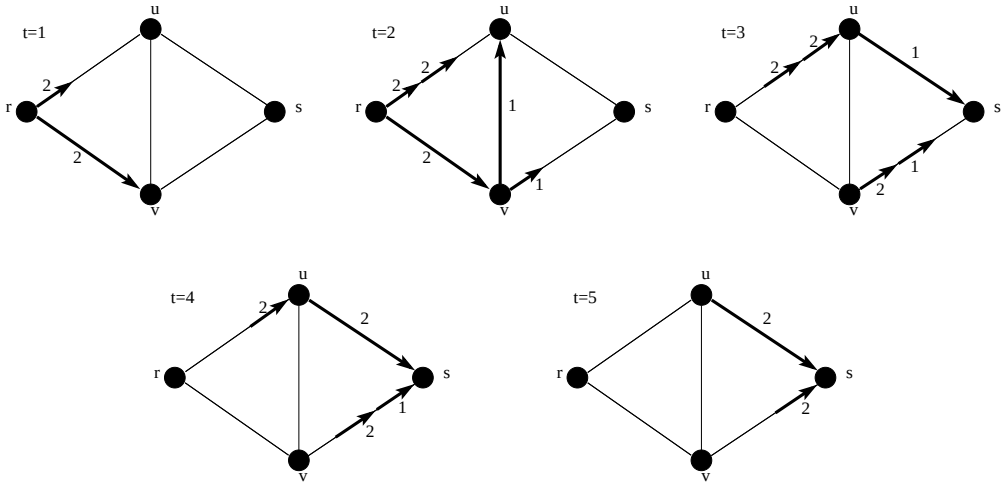

Figure 2: The dynamic flow at periods 1 to 5.

3.5 Formulations

Assuming fixed parameters and discrete time, the following constraints define the set of feasible dynamic flows.

$$\sum_{t=0}^{T}\left(\sum_{a\in\delta^+(r)}x_a(t-\tau_a)-\sum_{a\in\delta^-(r)}x_a(t)\right)=-d \tag{5}$$

$$\sum_{t=0}^{\theta}\left(\sum_{a\in\delta^+(v)}x_a(t-\tau_a)-\sum_{a\in\delta^-(v)}x_a(t)\right)\geq 0 \qquad \forall v\in V\setminus\{r,s\},\ \forall\theta=0,1\ldots,T-1 \tag{6}$$

$$\sum_{t=0}^{T}\left(\sum_{a\in\delta^+(v)}x_a(t-\tau_a)-\sum_{a\in\delta^-(v)}x_a(t)\right)=0 \qquad \forall v\in V\setminus\{r,s\} \tag{7}$$

$$\sum_{t=0}^{T}\left(\sum_{a\in\delta^+(s)}x_a(t-\tau_a)-\sum_{a\in\delta^-(s)}x_a(t)\right)=d \tag{8}$$

$$0\leq x_a(t)\leq u_a \qquad \forall a\in A,\forall t=0,1\ldots,T \tag{9}$$

In continuous time the sums over time are replaced by integrals. For example, in the continuous time model with varying arc and storage capacities the constraints are the following:

$$\int_{t=0}^{T}\left(\sum_{a\in\delta^+(r)}x_a(t-\tau_a)-\sum_{a\in\delta^-(r)}x_a(t)\right)\,\mathrm{d}t=-d$$

$$0\leq\int_{t=0}^{\theta}\left(\sum_{a\in\delta^+(v)}x_a(t-\tau_a)-\sum_{a\in\delta^-(v)}x_a(t)\right)\,\mathrm{d}t\leq b_v(\theta) \qquad \forall v\in V\setminus\{r,s\},\ \forall\theta\in[0,T)$$

$$\int_{t=0}^{T}\left(\sum_{a\in\delta^+(v)}x_a(t-\tau_a)-\sum_{a\in\delta^-(v)}x_a(t)\right)\,\mathrm{d}t=0 \qquad \forall v\in V\setminus\{r,s\}$$

$$\int_{t=0}^{T}\left(\sum_{a\in\delta^+(s)}x_a(t-\tau_a)-\sum_{a\in\delta^-(s)}x_a(t)\right)\,\mathrm{d}t=d$$

$$0\leq x_a(t)\leq u_a(t) \qquad \forall a\in A,\forall t\in[0,T]$$

Note that these formulations are not polynomial in the size of the input. The continuous time model has infinite number of constraints, the discrete one has more than T. The size of the input in this latter case is in the order of $\log(T)$, and with respect to this the size of the linear program is exponential. Thus, solving dynamic flow problems with linear programming is not a polynomial method.

4 Flow problems

The range of flow problems that can be defined on a dynamic network is obviously much wider than that on static networks. Besides meaningful flow problems in static networks (such as multicommodity, minimum cost or transshipment problems), there are problems where time is in question. In this section we deal with the main flow problems that can be addressed in a dynamic network, while in Section 5 we describe the solution methods proposed to solve these problems.

4.1 Maximum flow

The first and simplest dynamic model was introduced by Ford and Fulkerson [16, 17]. In their model the time is discrete and involved in only the traversal times of the arcs, as formulated by constraints (5)-(9). They dealt with the *maximum flow problem*, i.e., they maximized d; and gave a polynomial algorithm to get an optimal solution.

4.2 Quickest flow

The *quickest flow* problem asks for a flow of value d that finishes in minimal time. More precisely, it looks for the minimal T such that a flow with value d can be sent from the sink to the source in the time interval $[0, T]$. In some sense the quickest flow problem is the converse of the maximum flow problem, where we fix the time-span and maximize the flow. In discrete time the quickest flow can be found by a binary search on the time. Burkard, Dlaska and Klinz [8] gave a more efficient, strongly polynomial algorithm based on the discrete Newton's method. For continuous time, Fleischer and Tardos [15] proved that in case of integer d and integral traversal times, the time of the quickest flow is a rational number with a denominator bounded by the size of a minimum cut in the network. Thus, this time can also be found by binary search.

When we deal with the quickest dynamic flow, we have to differentiate between the *earliest* and the *fastest* solution. The difference is better explained through an example: Imagine that there are two bus lines that can take us to our destination. One runs frequently, but goes on a longer route. The other is an express line, takes much less time to reach the destination, but comes only a few times a day. We have two options, either we take the first bus, which will probably be the normal line; or we wait until the first express bus arrives. We will most probably arrive to our destination earlier if we take the first alternative, but the time we spend on the bus is less in the second case. The first is the earliest solution, the second is the fastest. More generally, the earliest flow is the one that accomplishes the task in minimum time, counting from zero. The fastest solution, on the other hand, starts counting the time when the first action takes place. If there is a temporary obstacle, then it might be worth beginning the task later, when the obstacle is lifted. If the network does not change, then there is no point waiting with the first action, and the fastest solution is the same as the earliest one.

One version of the quickest flow problem is to look for a *minimum delay* solution, which is a flow with the time spent by waiting at the nodes is minimized. Trivially, in the case where all traversal times are zero, the minimum delay flow is the quickest flow. The problem of minimum delay solution in continuous-time models with zero traversal times appears, for example, in [20, 31, 32, 39].

4.3 Minimum cost flow

If costs on the arcs are given, then one can fix the flow value d and the time limit T, and ask for the minimum cost dynamic flow that sends d amount of flow from the source to the sink in T time. A subproblem of this is the *minimum cost maximum dynamic flow problem* in which d is fixed to the maximum flow value in $[0, T]$, and the *minimum cost quickest flow problem* where T is the minimum time needed to send d flow. Klinz and Woeginger proved in [25] that both these minimum cost problems are NP-hard. Fleischer and Skutella [14] gave approximation results and algorithms for the minimum cost multicommodity flow problem.

4.4 Universally maximum flow

A flow that maximizes the throughput in time interval $[0, T]$ is not necessarily maximal for shorter periods. The dynamic network of Figure 1 on page 9 provides an example. (It is a slightly modified version of the example given by Ford and Fulkerson in [17].) The throughput up to $t = 3$ is 1 in the flow given in Figure 2. In Section 5.2 we will give another flow on the network with the same throughput in $\{0, 1, \ldots, 5\}$ but a throughput equal to zero up to $t = 3$. We will also show that both flows are maximal in $\{0, 1, \ldots, 5\}$, but obviously the second flow does not maximize the throughput in $\{0, 1, 2, 3\}$.

A flow that maximizes the throughput for every interval $[0, t]$, $0 \leq t \leq T$ is called the *earliest arrival flow*. Note that the literature is not consistent in the terminology, and the earliest arrival flows are often called universally maximum, as for example, in [11, 17, 18, 40]. Gale [18] showed the existence of such a flow for discrete time in a general dynamic network with time-varying parameters. Minieka [29] and Wilkinson [40] gave algorithms that find the discrete-time earliest arrival flow in a dynamic network with fixed parameters. Minieka's algorithms calls Ford and Fulkerson's algorithm for the maximum flow T times, while Wilkinson constructs a minimal cut. Neither algorithms are polynomial. There is no known polynomial algorithm for finding the earliest arrival flow. Hoppe and Tardos [23] gave a polynomial approximation. In continuous time, a theorem of Philpott [36] relates the earliest arrival flow to a minimum cost flow with a cost function that penalizes delays, thus proving the existence of an earliest arrival flow. Orda and Rom [33] give a construction for the earliest arrival flow in continuous time with piecewise continuous network parameters.

A close relative of the earliest arrival flow is the *latest departure flow* that maximizes the amount of flow leaving the source in every interval $[t, T]$, $0 \leq t \leq T$. A flow that is earliest arrival and latest departure at the same time is sometimes (e.g., in [15, 33]) called *universally maximum*. This causes an unfortunate confusion because, as noted above, the earliest arrival flow is sometimes also called universally maximum. We will differentiate between earliest arrival and universally maximum flows, keeping the latter for flows that are also latest departure. The algorithm of Minieka [29] gives the universally maximum flow in discrete time, while Fleischer and Tardos [15] shows that such a flow exists in continuous time. In both cases the network parameters are fixed.

4.5 Multiple terminals

Let us suppose that some nodes of the network are designated as terminals. A terminal can be either a source or a sink. We assume a priority order of the terminals and ask for a flow that lexicographically maximizes the amount of flow leaving each terminal in the given order. This is the *lexicographically maximum flow*. Note that it may happen that low priority terminals have negative or zero outgoing flow. A negative flow leaving a terminal means that positive flow enters it, so the terminal is in effect a sink.

We redefine the *transshipment problem* from Section 2.1. The network contains a set of terminals with supplies (positive if source, negative if sink). The aim is to find a flow that zeroes all supplies during the time horizon. It can be reduced to a lexicographically maximum flow problem both in discrete (Minieka [29] and Hoppe and Tardos [24]) and continuous time (Fleischer and Tardos [15]). Note that unlike in the static setting, the dynamic transshipment problem cannot be reduced to a single-source single-sink maximum flow problem. This is because the arc capacities in dynamic networks limit the flow rate and not the throughput. Therefore the capacity of the arcs connecting the supersource and supersink to the terminals cannot be set so that the total amounts of flow that pass through these arcs during the time horizon equal the supplies of the terminals.

The quickest transshipment, not surprisingly, is a transshipment that does the job in the shortest possible time. The special case where there is a single sink in the network is called the *evacuation problem*. The name comes from building evacuation, where one has to find the fastest way to transfer everyone from a building to the street. In telecommunications, a similar problem is to clear the network after a breakdown. Note that building evacuation is a good example for the need of an earliest arrival solution, because we want as few people to stay in the building as possible at every moment.

Hoppe and Tardos [23] provided a polynomial algorithm for the evacuation problem with fixed number of sources. They also gave the only known polynomial algorithm for the discrete quickest transshipment problem with arbitrary number of terminals in [24]. Fleischer and Tardos [15] extended this result for continuous time.

When the network graph is bipartite and all arcs connect a source with a sink, then we speak about the *transportation problem*. Bookbinder and Sethi reviewed the dynamic transportation problems in [6]. A shorter but more recent survey can be found in [28].

The universally quickest transshipment is a flow that minimizes the amount of excess left at the terminals at every moment in time. Hajek and Ogier [20] solve this problem polynomially when there is only one sink and all traversal times are zero. Fleischer [10] provides a two-source two-sink example for which a universally quickest transshipment does not exist, and presents a more efficient algorithm than that of [20] for the single-sink case.

4.6 Zero traversal times

One might ask the importance of dealing with dynamic networks with zero traversal times, as the main difference from static networks is that it takes time for a flow to traverse an arc. There is, however, an important additional feature of dynamic networks: the flow can be stored at nodes. Therefore it is possible to send more flow in a dynamic network with zero traversal times than the

maximum flow in the corresponding static network, by delaying some part of the flow at the source or at intermedial nodes. A typical question asked about a dynamic network with zero traversal times is that how much time is needed to send a given amount of flow that exceeds the static network capacity. Such questions were examined in [10, 11, 12, 20, 32]. Hajek and Ogier [20] gave the first polynomial algorithm for the quickest transshipment problem in a network with a single sink, zero traversal times and fixed arc capacities. Ogier [32] generalized this for piecewise constant arc capacities. Fleischer provides a faster algorithm for Hajek and Ogier's problem in [10], and for Ogier's problem in [11]. In the latter paper Fleischer formulates the problem as a parametric maximum flow problem (i.e., a static flow problem in which the arc capacities are functions of a parameter); and then modify the existing algorithms for parametric flows to solve the dynamic flow problem.

4.7 Varying parameters

Things get more complex when the parameters of the network, such as capacities or traversal times, are not fixed but they can change during the time horizon. For example, this is the case when the network models a real-world transportation system. Two main approaches are known to analyze these time-varying dynamic problems. The first assumes that the capacities are constant but the traversal times of the arcs change. Among the papers listed in the references, [26] and [27] follow this way. The other approach is when traversal times are fixed but the capacities vary, as in [2, 3, 22, 30, 36]. Orda and Rom [33] combine the two approaches and give theoretical results for continuous-time dynamic flows in networks where the capacities and traversal times change over time.

If traversal times change, then it is usually assumed that they are determined by the load on the arcs or by the flow rate at the time when the flow enters the arc. Köhler and Skutella [27, p. 2] discusses the difficulties related to this approach:

> The critical parameter for modeling temporal dynamics of time-dependent flows is the presumed dependency of the actual transit time τ_e on the current (and maybe also past) flow situation on arc e. Unfortunately, there is a tradeoff between the need of modeling this usually highly complex correlation as realistically as possible and the requirement of retaining tractability of the resulting mathematical program.
>
> Due to the latter condition, many models in the literature rely on relatively simple functions. For example, the transit time of an arc is often treated as a function of only the flow rate at time of entry to the arc. However, in many cases this assumption is unrealistic since it does, for example, not preserve the first-in-first-out property encountered in most applications.
>
> In contrast, a fully realistic model of flow-dependent transit times on arcs must take density, speed, and flow rate evolving along the arc into consideration [19]. Unfortunately, even the solution of mathematical programs relying on simplifying assumptions is in general still impracticable, i.e., beyond the means of state-of-the-art computers, for problem instances of realistic size (as those occurring in real-word applications such as road traffic control).

Köhler and Skutella [27] employ a model in which at each moment in time the uniform speed on an arc depends only on the load which is currently on the arc. They claim that this assumption captures the behavior of road traffic on short streets, and note that longer streets can be replaced by a series of shorter streets. They also prove that the problem of finding the quickest flow in this setting is strongly NP-hard, so cannot be approximated in polynomial time with arbitrary precision, unless $P = NP$. Nevertheless, they give a $(2 + \epsilon)$-approximation. Köhler, Langkau and Skutella [26] deal with a model in which the traversal time of a flow unit is fixed when it enters the arc (i.e., it is *inflow-dependent*). They give a $(2 + \epsilon)$-approximation for the time of the quickest flow.

As for the approach where the arc and storage capacities vary, in the most general setting they are assumed to be non-negative, bounded, Lebesgue-measurable functions. The papers [2, 3, 36, 33] deal with this model in continuous time and provide mainly theoretical results, such as the existence of the optimal solutions or the convergence of the algorithms that can eventually find them. Halpern [22] gives an algorithm for the discrete-time version. The algorithms provided in these papers are not efficient in practice and their implementations do not seem able to handle problems with more than a few nodes (as noted in [10]).

There are papers [11, 32] (and also [22, 30]) where the capacities are piecewise constant functions. As opposed to the general model, for piecewise constant capacities there are efficient algorithms to find an optimal flow (see [11]). However, these algorithms assume that all traversal times are zero.

4.8 Complexity

We summarize here the complexity results for the problems mentioned in this section. Polynomial algorithms exist for the maximum flow, quickest flow and quickest transshipment problems in both discrete and continuous time for fixed parameters. The minimum cost problem is NP-hard, a fully polynomial approximation scheme was given in [14]. For the earliest arrival flow problem, there is neither a polynomial algorithm known nor a proof of NP-hardness. The same applies for the universally maximum flow problem. As for multicommodity, Hall, Hippler and Skutella [21] recently proved that even the fractional dynamic multicommodity flow problem is NP-hard.

In models where the traversal times vary, we have only approximative polynomial results. The quickest flow problem with load-dependent traversal times is strongly NP-hard. The best known approximation has a ratio of $(2 + \epsilon)$. The same ratio was given for inflow-dependent traversal times.

Finally, a polynomial algorithm for a special case: In networks with zero traversal time and piecewise constant capacities, the quickest transshipment can be found in polynomial time.

5 Techniques

In this section we describe the main techniques developed for solving dynamic network flow problems. We focus on generic methods, that are applicable to a wide set of problems. Every specific question, of course, needs a tailored solution but the techniques given here can provide a good framework.

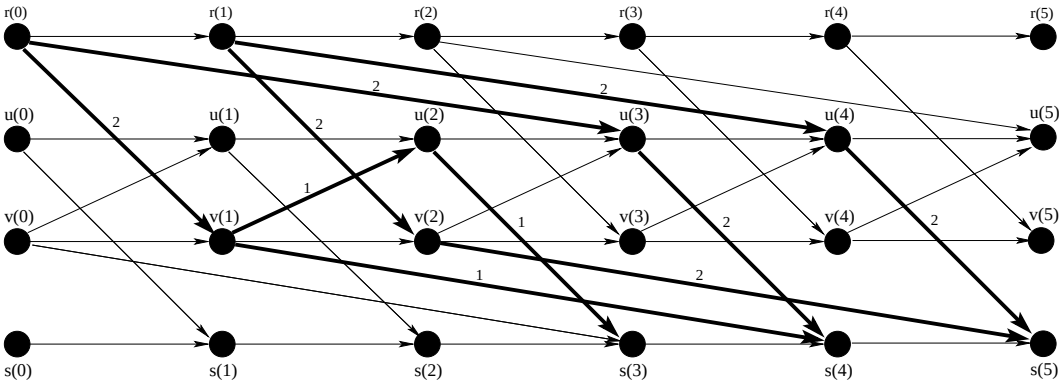

Figure 3: The time-expanded version of the dynamic network of Figure 1 and the equivalent of the flow given in Figure 2.

The methods used in discrete and continuous time are quite different. In general, there are more practical solutions for discrete time, whereas for continuous-time problems one can often find only theoretical results. The usual approach to give practical algorithms for continuous-time network problems is to reduce it to discrete time. The two ways of doing this are via the natural transformation or with discretization. In the natural transformation, described in more details in Section 3.2, we solve the discrete version of the problem and then prove that the optimal continuous solution can be achieved from the discrete solution by extending the flow values to the unit intervals separating the discrete time instances. Discretization works by choosing a suitable time unit and considering the continuous time as broken up to integer time periods. This approach loses on precision but gains on applicability.

5.1 Time-expanded graphs

The first solution technique for dynamic flow problems is due to Ford and Fulkerson [16]; later reviewed in their seminal book on network flows [17]. The main discovery was the *time-expanded graphs*. The time-expanded version of a dynamic network is a static network in which there is a copy of the nodes for each time step in the time horizon $\{0, \ldots, T\}$ and the arcs are redrawn between these copies to express their traversal times. Figure 3 gives the time-expanded graph for the network of Figure 1 for the time horizon $\{0, \ldots, 5\}$. The formal definition is the following. The time-expanded version of network G is a digraph G^T with nodes $v(t)$ for all $v \in V$ and $t \in \{0, 1, \ldots, T\}$; and for every arc $(u, v) \in A$ with traversal time τ_{uv} and capacity c the graph G^T has arcs $\big(u(t), v(t + \tau)\big)$ with capacity c for $t = 0, \ldots, T - \tau_{uv}$. There are also arcs $\big(v(t), v(t + 1)\big)$ with capacity b_v for all $v \in V$ and $t = 0, 1, \ldots, T - 1$ to represent storage at v. Note that if we use loops to model storage, we get these holdover arcs automatically from the general arc construction.

It can be seen quite readily that a dynamic flow on G from the source r to sink s is equivalent to a corresponding static flow on G^T from the source set $r(t)$ to the sink set $s(t), t = 0, 1, \ldots, T$. For example, the flow given in Figure 2 corresponds to the flow defined by the bold arcs in Figure 3. One can, of course, transform the static flow in G^T to a single-source single-sink problem by introducing a supersource and a supersink. Therefore, finding a flow in the dynamic network can be solved by finding a static flow in the time-expanded graph. The size of the time-expanded graph, however, is not polynomial in the size of the input, as there are T copies of the network. Consequently, any algorithm based on G^T cannot be polynomial. Ford and Fulkerson devised an algorithm that does not use the time-expanded graph to find a maximum flow in G, as explained below. They used G^T only to prove that the flow their algorithm finds is indeed optimal.

5.2 Temporally repeated flows

The basic idea of Ford and Fulkerson's algorithm is to find the maximum flow among the *temporally repeated flows*, defined as follows. Let $f : A \to \mathbb{R}$ be a feasible static flow in G, and $f^1, \ldots, f^k$ be a chain decomposition of f, i.e., P^i is a chain, f^i is the flow on it and $\sum_{i=1}^{k} f^i = f$. Let $\tau(P^i)$ denote the sum of the traversal times on the arcs belonging to P^i. Now for each $i = 1, \ldots, k$, send $|f^i|$ units of flow along P^i at every time step from 0 to $T - \tau(P^i)$. By this we get a dynamic flow f^T, the value of which is

$$|f^T| = \sum_{i=1}^{k} \big(T - \tau(P^i) + 1\big)|f^i| = (T+1)|f| - \sum_{a \in A} \tau_a f_a. \tag{10}$$

Thus the value of f^T does not depend on the chain decomposition chosen. Furthermore, finding a maximum temporally repeated flow amounts to finding a minimum cost circulation with arc costs τ_a and an additional supersink-supersource arc with cost $-(T+1)$ and infinite capacity. Ford and Fulkerson showed that there is always a maximum dynamic flow among the temporally repeated flows. The crux of their proof is finding a cut in the time-expanded graph that has the same capacity as the value of the maximum temporally repeated flow, and then evoking the max flow-min cut theorem for static flows. As a consequence, the maximum flow in this model can be calculated by solving a minimum cost circulation problem on the original network.

Take, for example, the network of Figure 1. For $T = 5$, the minimum cost circulation is given by two cycle flows of value 2: one is on the cycle $r \to u \to s \to r$, the second is on $r \to v \to s \to r$. The corresponding flow on the time-expanded graph is shown with bold arcs in Figure 4. The optimality of this flow (and that of the one given in Figure 3) can be checked by finding a cut of value 8 in the time-expanded graph.

The relationship expressed in (10) can be extended to continuous time. Anderson and Philpott showed in [3] that in case of fixed arc capacities the continuous maximum dynamic flow in time T has value $T|f| - \sum_{a \in A} \tau_a f_a$ where f is a minimum cost static circulation in the network with an additional sink-to-source arc with cost $-T$ and infinite capacity.

Temporally repeated flows are very useful to get the maximum dynamic flow. For other problems, however, they do not necessarily provide an optimal solution. We have seen an example in

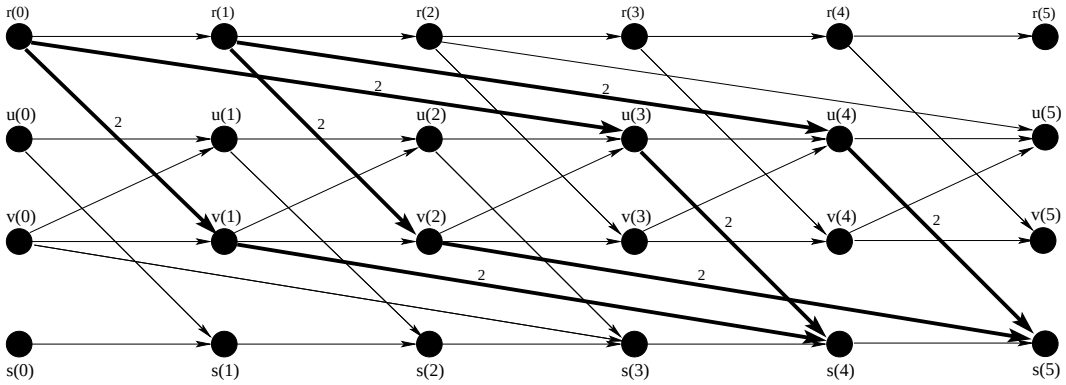

Figure 4: The maximum dynamic flow on the network of Figure 1.

Section 4.4 when the temporally repeated maximum solution was not earliest arrival. Klinz and Woeginger [25] gave an example of a network in which the set of temporally repeated flows does not contain the minimum cost flow. Hoppe and Tardos [24] introduced the *non-standard chain decomposition*, in which a chain flow may traverse an arc in opposite direction to the original flow. By this, they managed to devise polynomial algorithms for the quickest transshipment problem in discrete time.

5.3 Condensed time-expanded graphs

The time-expanded graph is clearly not a practical notion in continuous time because of the infinite number of time instances. A method to deal with this difficulty is to discretize the time with a sufficient roughness to get a *condensed time-expanded graph* of polynomial size. Note that changing the length of the time step can also be an approach to overcome the inherent exponential size of the time-expanded graph in discrete time. Rougher discretization, however, leads to less precise results. Philpott and Cradock [37] proposed an adaptive discretization algorithm that can choose the length of the times steps so that it provides the best performance. Fleischer and Skutella [13] devised a discretization method that provides satisfactory precision but keeps the complexity of the resulting time-expanded graph low. They showed that there is an appropriate choice of the time-step length that gives a polynomial-size condensed time-expanded graph and allows a $(1+\epsilon)$-approximation in time for the quickest flow or transshipment problems. In [14], Fleischer and Skutella improve their analysis and show that fewer time-layers are enough to obtain the same precision in approximation.

5.4 Averaging

Another, simpler approximation method was also given in [13]. This method is based on *averaging* the flow on every arc. That is, let x be a feasible dynamic flow rate in time horizon $[0, T]$, and y be

a static flow with the following values on the arcs:

$$y_a = \frac{1}{T}\int_0^T x_a(t)\,\mathrm{d}t \qquad \text{for all } a \in A.$$

This static flow is feasible, and its throughput during time horizon $[0, T]$ is the fraction $1/T$ of the throughput of x. Furthermore, y is *T-length-bound*, which means that it can be decomposed into chains such that the length of any chain (assuming the traversal times to be distances) is at most T. Finding a T-length-bound static flow is NP-hard, but can be approximated by a factor of $(1+\epsilon)$, as shown in [13]. The process of getting y from x can be reversed. Given a static feasible T-length-bound flow y with throughput d, we can obtain a feasible dynamic flow with throughput Td by pumping the chain flows determined by y into the network for T time units, and then wait for another T time units until all flow units arrive. This method allowed Fleischer and Skutella to get a $(2+\epsilon)$-approximation for the time of the quickest flow.

5.5 Cuts

The static flow algorithms are based on the famous max flow-min cut result of Ford and Fulkerson. A similar relationship were discovered for dynamic flows. In discrete time, cuts can be found in the time-expanded graphs, as it was already done by Ford and Fulkerson in [17]. The dynamic analogue of the cut is a set valued function $C : \mathcal{T} \to A$, such that at each time the value of C is an ordinary cut in the network. From a slightly different point of view, dynamic cuts change with time, arcs at some point can become members of the dynamic cut and remain there for a given period. This notion can be adapted to continuous time, as done by Philpott [36], following [2]. Generalized continuous dynamic cuts are used for getting theoretical results in [2, 3, 36]. On the other hand, Ogier [32] employed cuts to get a polynomial algorithm for the earliest arrival flow problem in continuous time with zero traversal times and piecewise constant capacities.

5.6 Alternative approaches

The multiperiod structure of a discrete dynamic flow problem can be exploited by a *forward algorithm*, a generic procedure for solving successively a multiperiod problem. Aronson and Chen [5] combined the forward technique with the network simplex method to get the forward network simplex method. This algorithm is applicable for discrete dynamic network flow problems that can be described with linear constraints and objective function. For more about forward algorithms and references, see the survey paper of Aronson [4].

Control theory is an alternative modelling approach for dynamic networks. It is more concerned with the change in the variables than their actual values. We define the states of the system and a control function that drives the system from the initial state to a desired final state via an optimal trajectory. The states of the system can represent the flows on the arcs or the storage at the nodes. The control function determines how the flow or storage changes in time. The optimality of a trajectory can be related to the value of the flow, its cost or to the time. Feasibility of a specific flow translates into the reachability of the final state. The control theory approach is more suitable for the

continuous-time problems, where the trajectory of the states can be described by the derivatives of the functions describing them. The basic reference of the application of control theory to dynamic flow problems is due to Moss and Segall [31]. Among our references, [20, 39] also use control theory.

Evolving graphs have been recently proposed by Ferreira [9] to model specific dynamic networks. An evolving graph is a sequence of subgraphs of a given underlying graph where each subgraph in the sequence represents a network topology existing for a given period of time. It can be argued that an evolving graph is equivalent to a dynamic network in which the network parameters, notably the arc capacities, are special piecewise-constant functions taking a value 0 if the subgraph does not contain the arc. Modelling such networks with evolving graphs provides a more compact description than with time-expanded graphs. This is especially true for continuous time, where time-expansion is not a viable option. Bui Xuan, Ferreira and Jarry [7] analysed evolving graphs, focusing on the least-cost path problems.

6 Annotated bibliography

This section lists most of the papers on which the previous sections are based. For each item we give some keywords to specify the problems attacked, then add comments which describe the main contribution of the paper. The items are ordered according to their subject.

Survey papers

Lovetskii, Melamed: Dynamic flows in networks [28]

A good survey with detailed description of the solution methods covered (e.g., control theory, time-expanded graph, labeling algorithms). Both discrete and continuous-time models. Problems include maximum flow, minimum cost flow, earliest arrival flow, maximum throughput, minimum cost transportation.

Aronson: A survey of dynamic network flows [4]

The paper is full of references for a wide range of problems and models, but concentrates mainly on the maximum flow and transshipment problems in networks with fixed traversal times in discrete time. The main advantage of this paper is the extensive coverage of applications.

Powell, Jaillet, Odoni: Stochastic and dynamic networks and routing [38]

It provides an extensive survey on dynamic and stochastic network problems (shortest paths, flows, etc.). The dynamic network part is very strong in modelling issues (especially about infinite horizons), not so much in techniques (mainly time-expanded graphs). It addresses only discrete-time models, but also deals with flow-dependent parameters.

Bookbinder, Sethi: The dynamic transportation problem: A survey [6]

A thorough description of the transportation problem both in discrete and continuous time.

Discrete time

Ford, Fulkerson: Flows in Networks [17] Maximum flow, discrete time, fixed traversal times and capacities.

The foundation of the static and dynamic flow theory. It introduces time-expanded graphs and temporally dynamic flows and proves that there is an optimal flow among the latter. Also gives a polynomial algorithm for the maximum flow problem in discrete time. Essential reference work.

Klinz, Woeginger: Minimum cost dynamic flows: the series-parallel case [25] Minimum cost problems, discrete time, fixed capacities and traversal times.

The paper proves that the minimum cost maximum dynamic flow problem and the minimum cost quickest flow problem are both NP-hard even for simple networks. A simple greedy algorithm is also given.

Hoppe, Tardos: The quickest transshipment problem [24] Quickest transshipment problem, discrete time, fixed traversal times and arc capacities.

Polynomial time algorithms for the lexicographically maximum dynamic flow problem and the quickest transshipment problem are given. The main idea is to introduce non-standard chain decomposition of flows in which the flow on a chain may traverse an arc in opposite direction to the one in the original flow.

Hoppe, Tardos: Polynomial time algorithms for some evacuation problems [23] Earliest arrival flow, evacuation problem, lexicographically maximum flow, discrete time, fixed parameters.

A preliminary paper to [24] above. A polynomial approximation of the earliest arrival flow, and exact algorithms for the evacuation problem and the lexicographically maximum flow are given.

Wilkinson: An algorithm for universal maximal dynamic flows in a network [40] Earliest arrival flow, discrete time, fixed parameters.

An algorithm is described to find the earliest arrival flow in a simple model. The algorithm is not polynomial.

Minieka: Maximal, lexicographic, and dynamic network flows [29] Earliest arrival flow, discrete time, varying parameters.

It provides results for static and dynamic flow, including the relation of the earliest arrival, latest departure and lexicographically maximum flows. For fixed parameters an algorithm (not polynomial) is given for the universally maximum flow.

Continuous time

Fleischer, Tardos: Efficient continuous-time dynamic network flow algorithms [15] Continuous time, fixed integral traversal times and arc capacities.

A very good overview of the existing results both for discrete and continuous time; and several new findings for continuous time, using the natural transformation of discrete flows to continuous ones.

Fleischer, Skutella: The quickest multicommodity flow problem [13] Quickest multicommodity flow, continuous time, fixed traversal times and arc capacities.

The introduction contains a very good survey of the existing results. They give a fully polynomial approximation scheme for the quickest multicommodity problem, i.e., a method that can provide a $(1+\epsilon)$-approximation for any ϵ. The main machinery is a condensed time-expanded network which relies on a rougher discretization of time. The paper also describes a simple $(2+\epsilon)$-approximation for the quickest multicommodity flow with costs based on averaging the flow.

Fleischer, Skutella: Minimum cost flows over time without intermediate storage [14] Minimum cost flow, continuous time, fixed traversal times and arc capacities.

It is shown that storage is not needed in the optimal solution. The authors give a strongly polynomial approximation scheme using a condensed time-expanded graph.

Hall, Hippler, Skutella: Multicommodity flows over time: Efficient algorithms and complexity [21] Multicommodity flow, continuous time, fixed parameters.

The dynamic multicommodity flow problem is proved to be NP-hard even for two commodities. For some special cases (e.g., for network with out-degree at most one) polynomial algorithms are given.

Philpott: Continuous-time flows in networks [36] Maximum flow, continuous time, varying capacities, fixed traversal times.

Philpott defines cuts in dynamic networks and proves a max flow-min cut theorem. The technique used is an analogue of the labeling algorithm of Ford and Fulkerson [17]. The paper also gives some results on the equivalence of the quickest maximum and the least cost evacuation problems.

Anderson, Nash, Philpott: A class of continuous network flow problems [2] Maximum flow, continuous and discrete time, varying capacities, zero traversal times on the arcs.

This is a special case of [36], above. Cuts are defined, and a max flow-min cut theorem is proved. The main theorem is erroneous, as pointed out in [36]. The techniques in [2] and [36] are very similar, but, of course, those in [36] are more difficult, and correct.

Orda, Rom: On continuous network flows [33] Maximum flow, continuous time, varying capacities and traversal times.

This paper proposes an extension of [36] to varying piecewise-continuous traversal times. The capacities are also piecewise continuous. A max flow-min cut theorem is given, as well as a construction of an earliest arrival flow.

Anderson, Philpott: Optimisation of flows in networks over time [3] Maximum flow, continuous time, varying capacities, fixed traversal times.

The paper explores some questions connected with duality theory, and the form of the optimal solutions.

Philpott, Craddock: An adaptive discretization algorithm for a class of continuous network programs [37] Minimum cost flow, continuous time, piecewise-constant arc capacities, piecewise-linear cost function.

An adaptive discretization method is presented and its performance is compared to other approaches.

Flow-dependent traversal times

Köhler, Langkau, Skutella: Time-expanded graphs with flow-dependent transit times [26] Quickest flow, continuous time, fixed arc capacities, inflow-dependent traversal times.

The main contribution is to introduce *fan graphs* and *bow graphs* to model the flow-dependency of the traversal times. Fan graphs give a precise model, but yield huge networks (basically a time-expanded graph for varying traversal time arcs), while bow graphs lead to a polynomial $(2+\epsilon)$-approximation.

Köhler, Skutella: Flows over time with load-dependent transit times [27] Quickest flow, continuous time, fixed arc capacities, load-dependent traversal times.

They give a $(2+\epsilon)$-approximation for the quickest dynamic flow using temporally repeated flows. The method is an application of the ideas of Ford and Fulkerson [17] and Fleischer and Skutella [13]. They also prove that the problem is strongly NP-hard, not being approximable in polynomial time with arbitrary precision, unless $P = NP$.

Varying capacities

Minieka: Dynamic network flows with arc changes [30] Maximum flow, discrete time, fixed traversal times, arcs can be added to or deleted from the network. No storage.

Ford and Fulkerson's algorithm is modified in this paper to handle networks with simple changes in the layout. The author has to assume that there is no storage to avoid difficulties.

Halpern: A generalized dynamic flows problem [22] Maximum flow, discrete time, fixed traversal times, varying arc capacities. Storage may not be allowed at nodes at specific times.

The paper gives a detailed augmenting path algorithm to solve the problem. Some words about complexity and a comparison with the time-expanded graph approach.

For more papers with varying capacities, see also [2, 3, 33, 36, 37] listed in other subsections.

Zero traversal times

Fleischer: Faster algorithms for the quickest transshipment problem [10] Quickest transshipment, discrete and continuous time, zero traversal times, fixed arc capacities.

It presents a polynomial algorithm that reduces the problem to a polynomial number of maximum flow computations and a faster algorithm that uses minimum cost flow computations. The main technique is a two-level network inspired by the time-expanded graphs. One level represents the first time unit, the other represents all other time units, and then this representation is repeated at most k times, where k is the number of terminals. The author also gives an algorithm to get integral flows. The continuous-time solutions are achieved via the natural transformation.

Hajek, Ogier: Optimal dynamic routing in communication networks with continuous traffic [20] Minimum delay flow, continuous time, zero traversal times, fixed arc capacities.

This paper uses a control theory approach, and gives a polynomial algorithm to find the optimal flow.

Fleischer: Universally maximum flow with piecewise-constant capacities [11] Earliest arrival flow, continuous time, zero traversal times, piecewise-constant capacities.

It is shown that the problem can be reduced to a generalized parametric maximum flow problem. Thus the paper provides a faster algorithm than that of [32].

Ogier: Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities [32] Earliest arrival flow, continuous time, zero traversal times, piecewise-constant integer capacities.

A polynomial algorithm for the earliest arrival flow is given. The author proves a max flow-min cut result and uses it in the algorithm. He also shows that the minimum delay flow problem is equivalent to the earliest arrival problem.

Fleischer, Orlin: Optimal rounding of instantaneous fractional flows over time[12] Minimum cost quickest integral transshipment, discrete and continuous time, zero traversal times, fixed arc capacities, fixed storage capacities can also be handled, the cost function is convex and piecewise linear.

The focus of the paper is on discrete time, but the results can be extended to continuous time with the natural transformation. The authors look for integral solutions, and their contribution is a rounding technique that transforms a fractional stationary transshipment into an integral one with the same cost.

See also [2].

A classification

The following table classifies the references according to the network model they use. On the one hand, we differentiate networks that do not change in time, i.e., the arc capacities and the traversal times are fixed, from those in which the parameters might vary. On the other hand, we classify models according to the traversal times.

	Zero traversal times	Non-zero traversal times
Fixed parameters	[10] [12] [20]	[13] [14] [15] [17] [24] [25]
Varying parameters	[2] [11] [32]	[3] [22] [26] [27] [30] [36]

7 Conclusions

We have seen that the theory of dynamic flows is powerful enough to satisfy the need for more realistic network models. For some problems this statement can be extended to the practice as well. Unfortunately, however, we are not at the point of having a perfect solution to the road traffic problem described in the introduction. As quoted in Section 4.7 from [27], the existing solution methods are either too complex for practical settings, or need to assume significant simplifying conditions. But this difficulty should not discourage further research. On the opposite, although an all-purpose dynamic flow solution method is unlikely to appear, there is a clear opportunity for model-tailored

approaches that can provide sufficiently good solutions to real-world problems. For example, there is a distinct need for algorithms that can find the maximum or quickest solutions in a network where the traversal times are fixed but nonzero, while the capacities vary in a simple way, for example they are piecewise-constant functions of time. Ogier [32], and then Fleischer [11] dealt with this problem in a simpler form, with zero traversal times. Such a model appears in practical problems that can be described with evolving graphs. The same model was analyzed by Halpern [22], but his algorithm is most probably not efficient enough for instances of realistic size. Another possible way of attack is from the method of Hoppe and Tardos [24] that is suitable for setting where the traversal times are nonzero but the capacities do not change. Is there a way of efficiently introducing simple changes in the capacities? The question is ripe for solution.

8 Acknowledgments

I am grateful to Afonso Ferreira for his help at all stages of this work.

References

[1] R. K. Ahuja, Th. K. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, New Jersey, 1993.

[2] E. J. Anderson, P. Nash, and A. B. Philpott. A class of continuous network flow problems. *Mathematics of Operations Research*, 7(4):501–514, 1982. See also the Erratum in *Math. Oper. Res.*, 8(3):478, 1983.

[3] E. J. Anderson and A. B. Philpott. Optimisation of flows in networks over time. In F. P. Kelly, editor, *Probability, statistics and optimisation*, Wiley Ser. Probab. Math. Statist., pages 369–382. Wiley, Chichester, 1994.

[4] J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.

[5] J. E. Aronson and B. D. Chen. A forward network simplex algorithm for solving multiperiod network flow problems. *Naval Research Logistic Quarterly*, 33(3):445–467, 1986.

[6] J. H. Bookbinder and S. P. Sethi. The dynamic transportation problem: A survey. *Naval Research Logistic Quarterly*, 27(1):65–87, 1980.

[7] B. Bui Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[8] R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR Methods and Models of Operations Research*, 37(1):31–58, 1993.

[9] A. Ferreira. On models and algorithms for dynamic communication networks: The case for evolving graphs. In *Proceedings of* 4^e *rencontres francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL'2002)*, pages 155–161, Mèze, France, May 2002. INRIA Press.

[10] L. Fleischer. Faster algorithms for the quickest transshipment problem. *SIAM J. on Optimization*, 12(1):18–35, 2001.

[11] L. Fleischer. Universally maximum flow with piecewise-constant capacities. *Networks*, 38(3):115–125, 2001.

[12] L. Fleischer and J. B. Orlin. Optimal rounding of instantaneous fractional flows over time. *SIAM J. on Discrete Math.*, 13(2):145–153, 2000.

[13] L. Fleischer and M. Skutella. The quickest multicommodity flow problem. In W. J. Cook and A. S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 36–53. Springer, Berlin, 2002.

[14] L. Fleischer and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of the 35th ACM/SIAM Symp. on Discrete Algorithms (SODA)*, January 2003.

[15] L. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1998.

[16] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[17] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[18] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:59–63, 1959.

[19] N. Gartner, N. J. Messer, and A. K. Rathi (editors). Traffic flow theory: A state-of-the-art report. http://www-cta.ornl.gov/cta/research/trb/tft.html, 1997.

[20] B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks*, 14:457–487, 1984.

[21] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. In J. C. M. Beaten et al., editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 397–409. Springer, Berlin, 2003.

[22] J. Halpern. A generalized dynamic flows problem. *Networks*, 9:133–167, 1979.

[23] B. Hoppe and É Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 433–441, 1994.

[24] B. Hoppe and É Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000.

[25] B. Klinz and G. J. Woeginger. Minimum cost dynamic flows: the series-parallel case. In *Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 329–343. Springer, Berlin, 1995.

[26] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs with flow-dependent transit times. In R. H. Möhring and R. Raman, editors, *Algorithms - ESA '02*, volume 2461 of *Lecture Notes in Computer Science*, pages 599–611. Springer, Berlin, 2002.

[27] E. Köhler and M. Skutella. Flows over time with load-dependent transit times. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 174–183. SIAM, 2002. A revised version is available from Skutella's homepage.

[28] S. E. Lovetskii and I. I. Melamed. Dynamic flows in networks. *Automation and Remote Control*, 48(11):1417–1434, 1987. Translated from Avtomatika i Telemekhanika, No. 11, pp.7–29, 1987.

[29] E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21:517–527, 1973.

[30] E. Minieka. Dynamic network flows with arc changes. *Networks*, 4:255–265, 1974.

[31] F.H. Moss and A. Segall. An optimal control approach to dynamic routing in networks. *IEEE Transactions on Automatic Control*, 27(2):329–339, 1982.

[32] R. G. Ogier. Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities. *Networks*, 18:303–318, 1988.

[33] A. Orda and R. Rom. On continuous network flows. *Operations Research Letters*, 17:27–36, 1995.

[34] J. Orlin. Maximum-throughput dynamic network flows. *Mathematical Programming*, 27(2):214–231, 1983.

[35] J. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9(2):190–207, 1984.

[36] A. B. Philpott. Continuous-time flows in networks. *Mathematics of Operations Research*, 15(4):640–661, 1990.

[37] A. B. Philpott and M. Craddock. An adaptive discretization algorithm for a class of continuous network programs. *Networks*, 26:1–11, 1995.

[38] W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. et al. Ball, editor, *Handbooks in Operations Research and Management Science – Network Routings*, volume 8, chapter 3. Elsevier Science, 1995.

[39] G. I. Stassinopoulos and P. Konstantopoulos. Optimal congestion control in single destination networks. *IEEE Transactions on Communications*, 33(8):792–800, 1985.

[40] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifi que
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399